AN ENHANCED BROADCAST CHAIN INTEGRATION PROTOCOL

Jay Weber, Ceri Jerome, Prasad Panchangam, Daniel Salo
RespondTV, Inc.

Abstract

Operators and Programmers each have important roles in the delivery of enhanced content, and these roles need technical integration to make ETV work. In particular, programmers are in a position to determine the scheduling of enhanced content and other interactions with programming, and operators are in a position to determine bandwidth provisioning and platform-specific encoding. This paper describes an architecture and a protocol that supports integration of these control points along the broadcast chain.

INTRODUCTION

In this paper, *Enhanced Television (ETV)* is the combination of broadcast programming and synchronized interactive graphics, text, and forms. This category covers such compelling applications as direct response from advertisements, engaging product placements, audience participation, and rich T-commerce. Thus as ETV is a synchronization of programming and interactivity, it requires technical integration of the broadcast signal and the interactive capabilities of distribution and in-home equipment, in particular between programmers and operators.

Many pilots, including some of ours, have assumed that a programmer does all preparation of the signal and the Operator serves as a pass-through, perhaps with signal format conversion. We have found that this approach does not scale for the following two reasons.

First, Operators are deploying a variety of set-tops that have a diverse set of capabilities and a combination of standardized and proprietary interfaces. Programmers, who generally strive for maximum distribution, cannot prepare the interactive signal components in one way that fits all.

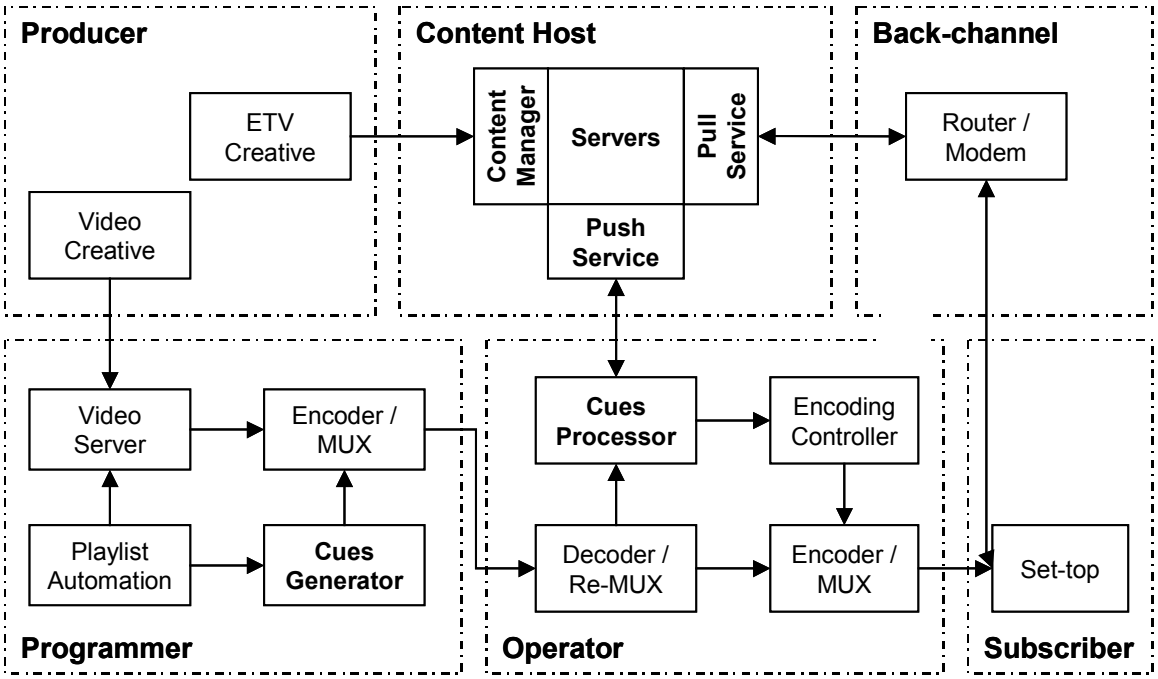
Second, distribution plants have varying requirements and capabilities in forward- and back-channel bandwidths and latencies. A basic choice is whether to embed content in the broadcast signal or send a trigger to load the content over a back-channel; the optimum choice varies by operator and over time even for a single operator.

To address this, we have developed a *Transport Control Architecture (TCA)* where the operators play an active role in managing how the broadcast signal directs content to subscriber equipment. Under the TCA programmers do not send instructions directly to set-tops but rather they send information, or *cues,* that operators use to control their set-tops.

This paper summarizes the TCA and then focuses on its key interface, the protocol for programmers to communicate ETV cues to operators.

TRANSPORT CONTROL ARCHITECTURE

In this paper, we use the term *programmer* to indicate any party that schedules video assets into a channel, and the term *operator* for any party that distributes such channels to consumer equipment. The architecture generalizes to situations where programmers and/or operators are one and the same, or multiple cascading entities such as station groups or meta head-ends.



The figure shows a simplified broadcast chain and back-channel, and the entities that constitute them. Note that the ETV Creative goes not to the programmer but to a *content host* who serves as a clearinghouse for multi-platform content and a server both for operators to load and cache content to be pushed over broadcast, and for set-top back-channels to pull personalized and transmit transaction data over back-channels.

The key components in the figure are the *cues generator* at the programmer and the *cues processor* at the operator. Cue generation takes information from play-list automation and encodes it for use downstream by operators. Cue processing reads such cues and acts upon them, usually by loading and caching content for data carouseling or transmitting a trigger.

The figure shows that the transport of cues is through an embedding in the broadcast signal. Though we anticipate that this will usually be the most convenient means, as we discuss later in this paper, other means are possible and have some advantages.

CUE SYNTAX

Cues may be transmitted using the following general format:

```
!!<URI>{[attr1:value1][attr2:value2]…
[attrN:valueN]}{[checksum]}
```

The prefix "!!" indicates the following characters represent a cue. The receiver may choose to act on the cue or to discard it. The remainder of the cue is designed to follow the syntax of triggers from EIA-608B.

The URI identifies the enhancement resource. The URI is enclosed in angled brackets, followed by zero or more pairs of attributes and values in square brackets, then followed by an optional checksum in square brackets. The order of attributes is not important.

As in EIA-608B, the characters included in the cues are interpreted as ISO-8859-1 (Latin-1) characters. Only those within the range 20h to 7Eh are used, the remaining characters should be discarded when processing cues.

Some URIs, attributes or values may require characters not within the above range, or that are excluded in this specification (such as square brackets). These characters should be encoded using the standard mechanism of "%" followed by the two-digit hexadecimal value in ISO-8859-1.

URI Field

The URI identifies a generic enhancement resource that is to be acted upon. An enhancement resource may be a single item such as a trigger, a content package, carousel file or schedule; or it may be a group of several such components.

In this context, a schedule refers to a set of enhancements and their timings throughout a television show or spot. A single cue is able to start the schedule, and the operator takes the responsibility of sending the enhancement components for the show as dictated in the schedule.

The URI can be specified as relative or absolute. Relative URIs allow operators to choose from where to obtain the enhancement data. Allowing the operator to choose the hosting service is a more flexible system, and can help with such issues as load balancing. An enhancement service provider may provide a web interface for obtaining the data; the details of this mechanism are described later in this document.

Attributes

Nine attribute fields are defined in this standard: *component, action, duration, offset, id, expires, source, utc,* and *response*.

The *component* attribute is used when the URI references a resource group; the value specifies a component within the group. The URI itself could contain component name, for example, the following cues identify the same resource:

```
!!<adsponsor/enh01>[component:content01]
!!<adsponsor/enh01/content01>
```

Keeping the URI as the root of the group (in this case, `adsponsor/enh01`) enables operators to group cues that apply to the same enhancement. This may be useful when applying actions to the whole group, such as *stop*.

The *action* attribute specifies how the resource is to be processed. The following table defines the current list of actions:

Action value	Action to take
start	Start the enhancement
stop	Stop the enhancement. If the URI specifies a group, stop all its components.
pause	Pause the enhancement and freeze the duration countdown. If the URI specifies a group, pause all its components
resume	Resume the enhancement after a pause.
load	Pre-load the enhancement
cancel	Cancel all previous matching cues. See "id" attribute for details.
query	2-way protocol: Ask for the status of an enhancement
response	2-way protocol: Reply to a status request

If an action is not specified, the default action is *start*.

The *duration* attribute is used with the *start* action. It specifies how long in milliseconds the enhancement should run for. A value of zero indicates the enhancement should run once only. If no duration is specified, the enhancement is run until a *stop* action is called.

The *offset* attribute specifies when the action should be executed. If a *utc* value is specified, the offset is relative to this time, otherwise the offset is relative to the time when the cue was received. Units are milliseconds. If no offset is specified, an offset of zero is used by default.

The *id* attribute, together with the URI and component attribute should uniquely

identify a cue. If a cue is sent multiple times, and these fields remain the same, the previous cues are discarded. For example, a cue with an offset of 30 seconds is sent every 10 seconds, with the offset value decreasing accordingly. The *id* will be the same each time, so the cue is only processed once. Multiple cues sent this way enable systems to cope with occasional delivery failures.

When a *cancel* action is processed, the results depend on whether the three identification fields are specified. If just the URI is specified, all cues with the same URI are cancelled. If the URI and component are specified, all cues matching both fields are cancelled. If the *id* is specified, only those cues matching all three fields are cancelled.

The *expires* attribute specifies in UTC when the cue is no longer valid. Expired cues should be discarded.

The *source* attribute specifies who sent the cue. The will usually be the programmer, but may also be the operator replying to a query in the 2-way protocol.

The *utc* attribute is a UTC timestamp for the cue. The offset attribute is relative to this value.

The *response* attribute is part of the 2-way protocol and contains the response message requested by a query action. If this attribute is present, the action attribute must be *response* and so is ignored.

Checksum

The checksum may be included to detect data corruption. It is identical to the Internet Protocol checksum described in IETF RFC 791 and IETF RFC 1071, also used in EIA-608B. The "!!" is included in the calculation. The result is a 16-bit value transmitted as four hexadecimal digits, with the most significant byte first. Characters outside the 20h to 7Eh range are not included in the checksum.

Attribute/value abbreviations

Cue transmission mechanisms often have bandwidth constraints that must be adhered to. The following policy is recommended to minimize bandwidth:

- Remove any attributes that would be present by default. This includes `[action:start]` and `[offset:0]`
- Use abbreviations for attributes and action values as shown in the table:

Attribute name	Abbreviation
component:xxx	c:xxx
action:start	a:s
action:stop	a:o
action:pause	a:p
action:resume	a:r
action:load	a:l
action:cancel	a:c
action:query	a:q
action:response	a:e
duration:mmm	d:mmm
offset:mmm	o:mmm
id:xxx	i:xxx
expires:ttt	e:ttt
source:xxx	s:xxx
utc:ttt	u:ttt
response:xxx	r:xxx

- Keep the URI and component attribute values to a minimum
- Use short id values
- Use short source values

TWO-WAY PROTOCOL

The protocol does not specify a delivery method or platform for the cues. Methods may include delivery in the T-2 service of a video broadcast, conforming to EIA-608 (Recommended Practice for Line 21 Data Services); or out of band delivery via TCP/IP or UDP/IP. The cues do not interfere with, and may be used in conjunction with triggers as in EIA-608B.

The one-way protocol is designed for in-band delivery of cues. It is assumed that in-band cues are synchronized to the broadcast, so that time offsets will be relative to the point in the broadcast when the cue is received.

An out-of-band delivery system may make use of the 2-way version of the protocol. This is an extension to the standard protocol adding the two actions *query* and *response*.

A programmer sends a query action cue asking for the status of an enhancement. The enhancement is identified by the URI, and optionally the component attribute. The operator replies with a response action cue, containing the status of the enhancement in the response attribute. The format of the response value is not defined by this standard and should be defined by the both the programmer and operator involved.

A query action cue containing an id attribute is a request for the status of a cue. The id, URI and component are used to match the cue. The operator replies with a response action cue containing the status of the cue in the response attribute. Again the format of the response value is not defined in this standard.

As mentioned before, a UTC time stamp can be included in cues that are not synchronized to the broadcast. If possible, this timestamp should take into account the time difference between the video feed at the programmer, and the video feed received by an operator. Alternatively, an operator could use the source field to identify the programmer, and adjust the timestamp accordingly.

SCHEDULE RESOURCE FORMAT

A schedule resource specifies a set of enhancement resources with a schedule of when each resource should be processed. The timings will be relative to a point in time, which is usually the beginning of a television show or spot.

A schedule should be specified as a list of cues. All the standard attributes apply except *source* and *utc*. The original cue that referenced the schedule resource determines both these attributes. Also, its offset value will be added to the offset value in each schedule cue.

CUE EXAMPLES

The following are all examples of valid cues:

```
!!<http://itv.adsponsor.com/spi/enh01
>[component:content03][action:start][
duration:30000]

!!<adsponsor/enh02>[source:aprogramme
r][expires:20010601][offset:10000][id
:0077]

!!<adsponsor/enh03/trigger01>

!!<adsponsor/enh04>[c:content02][a:l]
[s:aprogrammer][u:20010101T103922]

!!<adsponsor/enh04>[c:content02][a:q]
[s:aprogrammer]

!!<adsponsor/enh04>[c:content02][r:ac
tive][s:anoperator]
```

CUE PROCESSING

This section suggests some procedures that an operator should follow when processing cues.

A local cache is recommended for storing enhancements before insertion. The *load* action indicates enhancements are to be prepared in advance.

A content host's Service Provider Interface (SPI) is a convenient mechanism for supplying enhancement resources. The operator makes a request to the content host specifying the URI, component, and platform. The content host searches for the resource that matches the platform, and returns the required data. A platform may have several possible data formats, so the SPI should also return the specific data type. The cue only specifies the resource identifier; no assumption can be made about the generic data type (such as trigger or content) until the data type is returned by the SPI.

To summarize the process: the cue identifies the generic resource; the operator specifies the platform and requests the resource from a SPI; the SPI returns the resource and its data type.

Each enhanced television platform requires insertion tools and equipment to insert the data into the video stream. There may be several such systems available for each platform. The operator must retrieve the platform specific data and, if necessary, process the data so that it conforms to the insertion system interface.

A content host has the choice of supplying basic platform specific data, or supplying data that conforms to a specific insertion system. If an insertion system becomes standard for a platform then it makes sense for the service provider to provide data in the correct format. This would allow extra control over insertion parameters.

For standard television transmission, automation systems produce *as-run* logs to record the shows/spots that have aired. In the same way, operators should be able to produce as-run logs for enhancements that have been inserted. Each source will have its own associated log.

An operator and the programmers need to specify an error handling policy for the operator to implement. At the very least this should include steps to deal with cue parsing errors, data retrieval errors, and insertion errors. In the future the cue protocol may be extended to include an attribute to specify error-handling policy.

CONTENT HOSTING GUIDELINES

By providing a standard interface, content host service providers are able to serve content for multiple platforms from a single source. The suggested format of this interface is as follows:

ContentHostURL/EnhancementURI/Component

Where *Component* is optional. For example,

```
http://itv.contenthost.com/spi/adspon
sor/enh01/content01
```

This SPI format allows the operator to select the content host when the enhancement URI is relative, and also allows the programmer to specify the content host with an absolute enhancement URI.

The following table lists the parameters that should also be included in the request. Note that *platform* is the only required parameter.

Parameter	Value
`source`	Programmer; same value as cue source attribute
`client`	Operator
`platform`	Standardized name of the enhanced television platform
`device`	Standardized name of the insertion system
Any other platform- or insertion-specific parameters	

The content host should attempt to return data that matches the device. If the device is not specified, or is not recognized, the platform should be used to locate the data. If the platform is not supported or recognized, the request should be rejected.

The content host is also able to choose resources based on the programmer and operator. This may be useful for regional targeting of enhancements.

If an enhancement is not available for a platform, the contest host has the option of returning a default enhancement.

CONCLUSIONS

This paper has presented a new way for programmers and operators to synchronize in the deployment of Enhanced Television applications. We believe that it is a big win for the operators as it gives them the means to optimize for the network and their choices of delivery platforms. We also believe that it is a big win for the programmers who can achieve maximum ETV distribution with a minimum of distribution-dependent signal-processing.

While we have designed this protocol with some accounting for deployability, we recognize that there are several major open issues, including the following. First, as we mentioned the transport of the one- and two-way protocols between programmers and operators may depend on security and provisioning restrictions, as well as video formats. Second, that operators may need to make restrictions on the use of cues, e.g., that cues must appear at least some offset (as much as a day) before the interactive session.

Third, that this protocol must be more completely reconciled with existing and ongoing standard efforts. And finally, that resolution of the above issues will require bilateral and multilateral agreements among programmers and operators, and at some level this will require an accounting of larger business interests.

REFERENCES

EIA/CEA-608B, Line 21 Data Services (October 2000)

IETF RFC 1071, Computing the Internet Checksum (September 1988)

IETF RFC 2396, Uniform Resource Identifiers (URI): Generic Syntax (August 1998)

ISO-8859-1: 1987, Information processing – 8-bit single-byte coded graphic character sets – Part 1: Latin alphabet No. 1

Systems of Time,
http://tycho.usno.navy.mil/systime.html

CONTACT

Dr. Jay Weber
CTO
RespondTV, Inc.
1235 Howard St, San Francisco, CA 94103
jay@respondtv.com